Dear CS124 Grader,

About the Dataset and City Class

This dataset is a collection of the largest cities in the US by population. Each city by default is listed in order from the greatest population to the least, however for this project the cities were sorted in alphabetical order so that the population would not be in order from the first city to the last. This data was collected in 2013, and is one of the more recent data sets available already in the form of a .txt or a .csv file. This data is also very interesting. Cities are constantly growing and shrinking at the same time, especially these large cities. Each entry has 5 attributes, including the city name (string), the state of the city (string), the population of the city (integer), the rank of the city in terms of population (integer), and the percent growth in terms of population from 2000 to 2013 (double). Each of these attributes are fields of my City class.

There are getters and setters because there is no sensitive data. There is no input checking, because all data is perfectly sorted into a combination of strings, integers, and doubles.

The << operator has been overloaded, so each city can be listed in a consistent way in the console. The >, <, >=, <=, and == operators have also been overloaded so a City can be compared to another City.

Results of Separate Chaining and Linear Probing

Below are graphs of the results, which are also all listed out below in tables.

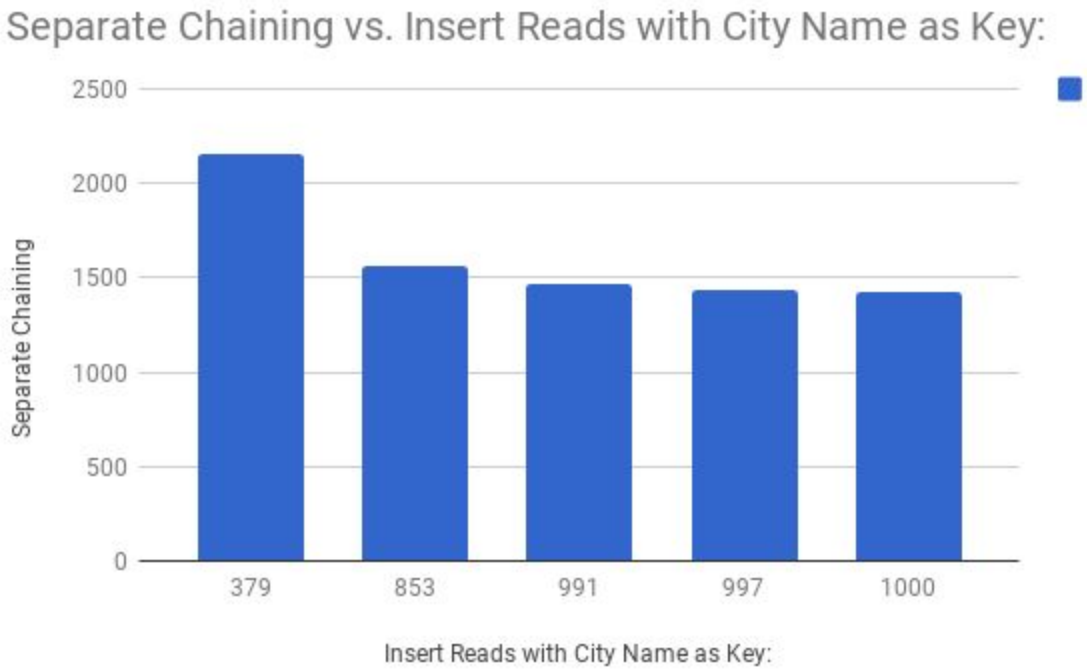
Separate Chaining	
Insert Reads with City Name as Key	
Table Size	Insert Reads
379	2152
853	1561
991	1463
997	1432
1000	1419
Average: 1605.4	

Separate Chaining	
Insert Reads with City Name as Key appended with string "APPEND TO STRING" (More Characters in Key):	
Table Size	**Insert Reads**
379	2162
853	1542
991	1511
997	1468
1000	1457
Average: 1628	

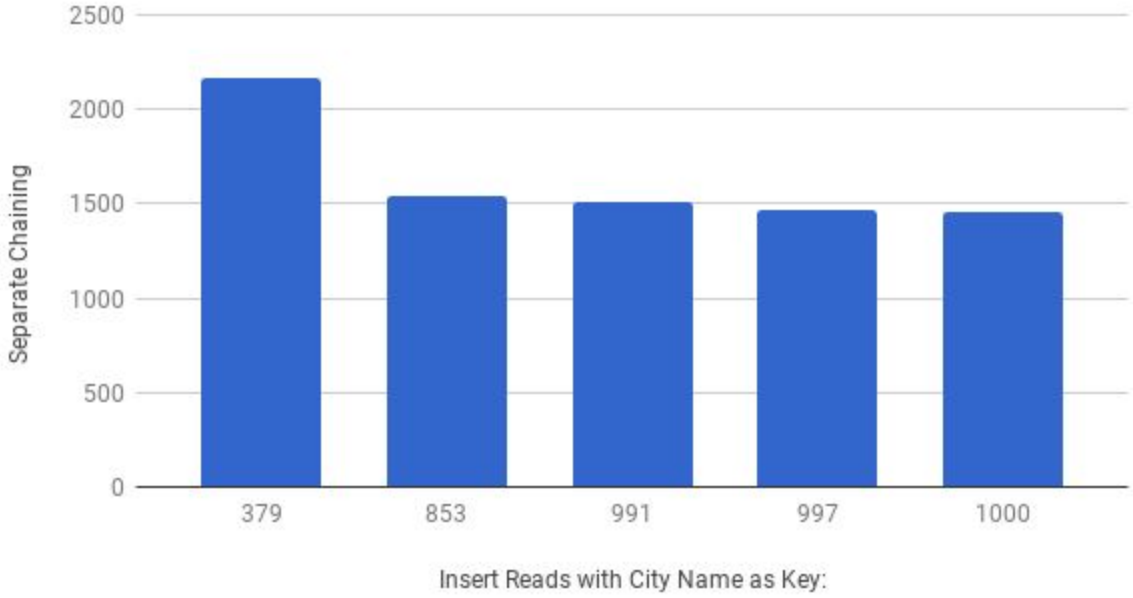
Linear Probing	
Insert Reads with City Name as Key	
Table Size	**Insert Reads**
379	8194
853	7747
991	4340
997	4010
1000	3873
Average: 5632.8	

Linear Probing	
Insert Reads with City Name as Key appended with string "APPEND TO STRING" (More Characters in Key):	
Table Size	**Insert Reads**
379	7969
853	7653

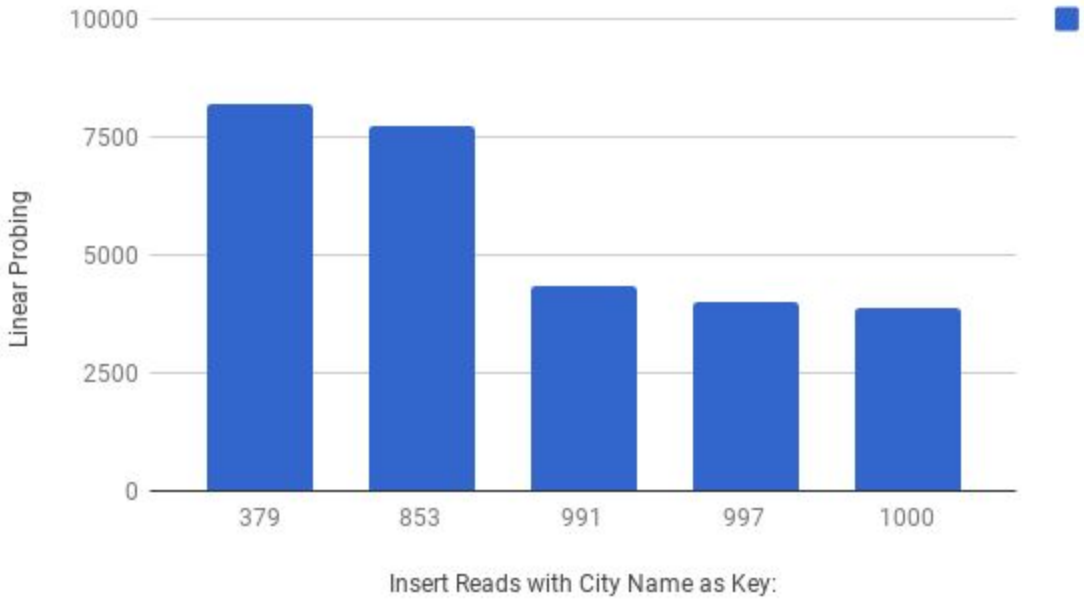
991	4265
997	4037
1000	3873
Average: 5559.4	

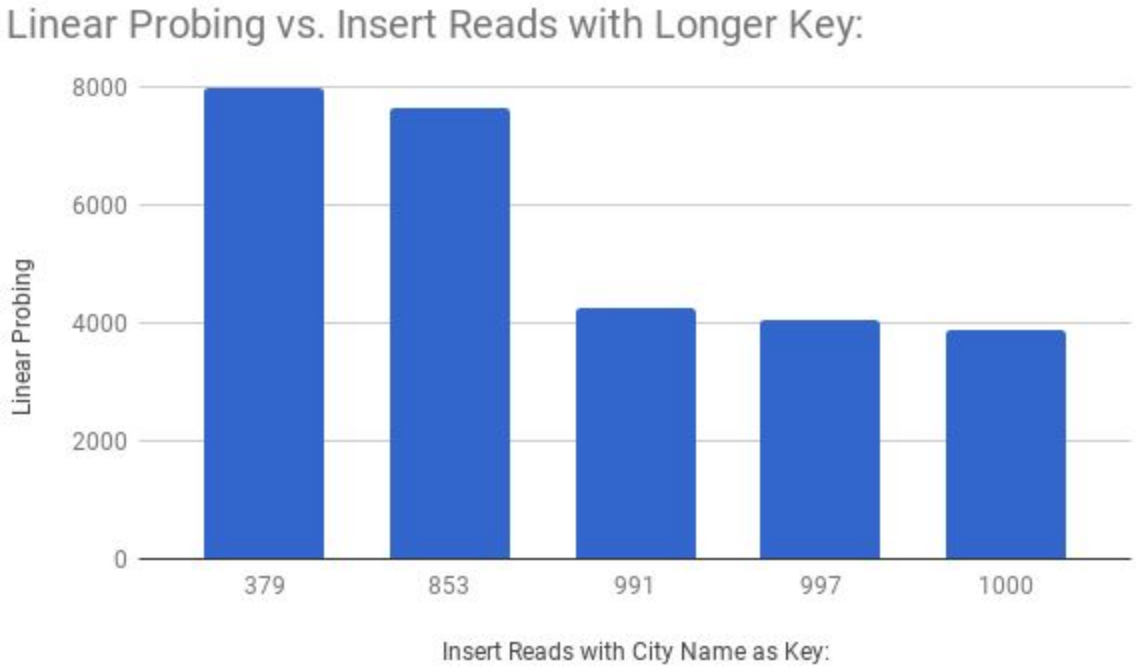


Separate Chaining vs. Insert Reads with Longer Key:



Linear Probing vs. Insert Reads with City Name as Key:





About the Results

Key Length:

After adding to the key, it was clear that the elements were being placed later in the hash table, due to a longer key. This resulted in slightly higher reads.

Removing, Searching:

Removing and searching the tables were both significantly lower in reads than insertion. This has to do with the fact that remove and search are both called many times when inserting. These two processes on their own are much less complex. Removal will usually only be used upon individual call, and search is used in insertion. It is used frequently and is essential to insertion.

Separate Chaining Vs. Linear Probing

In the case of this specific data set, separate chaining would be ideal. Since the size of the table can be as large as needed, linear probing will require more reads. This larger table size will act as a benefit for separate chaining because less reads are needed to search through shorter vectors.

Code Source

All code was taken from lecture materials from instructor Lisa Dion.